LLNL-TR-660035

Developing a User-Driven Framework for Generating Field Data Collection Applications

T. Lee

September 9, 2014

Developing a User-Driven Framework for Generating Field Data Collection Applications

Timothy Lee

Lawrence Livermore National Laboratory
Environmental Restoration Department
Livermore, CA 94550, USA
tdlee1230@gmail.com

Abstract— **This paper describes the implementation of a web-based framework, which allows end users to build custom data collection applications. The emphasis of this project is to ease the transition from handwritten forms to electronic mobile applications for data collection.**

Keywords— web-based, browser-based, mobile, data collection

I. Introduction

The Environmental Restoration Department (ERD) at Lawrence Livermore National Laboratory (LLNL) is tasked with investigating and cleaning up soil and ground water contaminated by past activities of LLNL and its predecessors at its Livermore and Site 300 locations. ERD develops and applies interdisciplinary science and innovative technology to restore the environment and reduce the impacts of environmental insults [2]. One of the methods in which ERD measures the effectiveness of their remediation strategies is by periodically drawing water samples from nearly 2000 wells they have drilled in and around LLNL's sites. The measurements confirm that ERD's efforts have been successful over the course of the past 35 years. However, when samplers collect data from the wells, they print out a group of forms that they bring into the field and manually record their measurements. This method of data collection raises two concerns: the ongoing use of paper, and the possibility of paper-to-database transcription errors.

For these reasons, ERD is looking to transition towards a mobile method of collecting data. However, this idea has three concerns: there are many forms for which mobile applications would need to be created, the mobile product must be deployable on different devices, and the mobile product must be deployable in areas that may not have a reliable Internet connection, such as some of the wells at Site 300. To address these concerns, ERD proposed the idea to construct a framework which dynamically creates browser-based mobile applications to fill samplers' specific needs. For my summer internship, I was assigned to assist in developing this project.

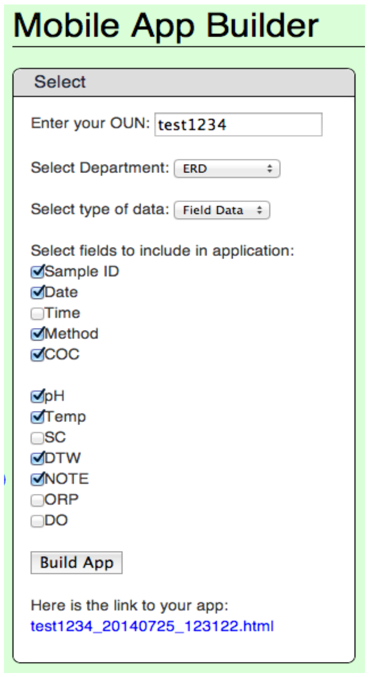
In this paper, the focus will be on the progress I have achieved on the framework under the guidance of ERD. The paper will discuss the framework's implementation, the primary goals it accomplishes, and how it will impact LLNL when it is finished.

II. Background

Browser-based applications are becoming popular due to their cross-platform capability and relative ease of development. Developing applications native to a specific operating system (OS) requires extra knowledge such as understanding the language the OS uses (along with its specific quirks) and being familiar with the development kits for the OS. Development becomes increasingly burdensome when deploying a natively-built application on multiple OSes. On the other hand, almost all modern mobile devices have a web browser pre-installed on them. This opens opportunities for developers to capitalize on one of the few things all of those devices have in common. Web development is a very broad and well-documented field, which makes developing applications extremely easy. The trade-off from using web-based design over native design is that web-based applications work on any device with a web-browser, but their user interface (UI) is typically more challenging to create so that it fits on all devices [1]. Furthermore, web-based applications have limited capabilities compared to native applications. However, the capabilities of web-based applications are sufficient for ERD's needs. For this framework, the trade-off works in ERD's benefit - if the data collection applications were web-based, they would work on any web-enabled device, saving data collectors from having to invest in large numbers of specific devices. Furthermore, the forms are simple enough that intuitive layout and interface design is fairly easy to accomplish.

III. Product Description

A. The Framework


Mobile App Builder

Select

Enter your OUN: test1234

Select Department: ERD

Select type of data: Field Data

Select fields to include in application:
- ✓ Sample ID
- ✓ Date
- ☐ Time
- ✓ Method
- ✓ COC
- ✓ pH
- ✓ Temp
- ☐ SC
- ✓ DTW
- ✓ NOTE
- ☐ ORP
- ☐ DO

Build App

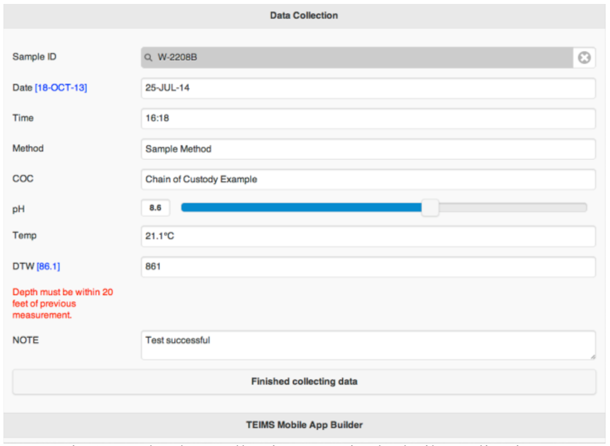
Here is the link to your app:
test1234_20140725_123122.html

Fig 1. The application builder.

The framework is an application builder which is implemented using jQuery, Amplify, and Perl. jQuery is a powerful, commonly-used JavaScript library which helps make developing responsive, interactive pages easier. Amplify is a JavaScript library that provides an abstraction interface that makes it easier to manage local storage of data on a device. The user is presented with inputs for entering their user name, department, and type of data to collect. Based on these choices, the menu dynamically updates using jQuery to show a list of checkboxes that correspond to fields the user would like to have in his or her application. Once all of the choices have been made, the user clicks the "Build App" button, which instructs the Perl handler to generate the files for the application. Each checkbox that is checked to be included in the application passes some data into the handler. The handler then iterates through all of the checked items and creates UI components for each of them. The application supports autocompletion and input validation. If one of the checked boxes has data to pass from the database to the built application, Amplify will handle storing it on the device on which the user is building their application. The application builder produces two files: an HTML file which lists the libraries the application will need, and a JavaScript file which contains all of the application's layout and features.

B. Generated Application

The built application is coded using four JavaScript libraries: jQuery Mobile, React, jQuery Validate, and Amplify. jQuery Mobile and React are the primary libraries in displaying the layout, while jQuery Validate and Amplify add extra functionality to the application. jQuery Mobile is a UI library which abstracts all of the work of making the application fit on all browser sizes. React is a layout framework library, designed to wrap sections of HTML code into components, which can be called using convenient single-line functions. The key feature of React is that components can be very specific, such as wrapping a text input and its accompanying label into a component; but, they can also be very broad, such as wrapping the contents of a page into a component. This flexible methodology allows for an organized hierarchy of broad-to-specific components that accurately models how a UI is designed. jQuery Validate is a plugin for jQuery, which makes error checking very simple. Lastly, Amplify is used here to retrieve the local data stored by the application builder and store the collected data for uploading.


Data Collection

Sample ID	W-2208B
Date [18-OCT-13]	25-JUL-14
Time	16:18
Method	Sample Method
COC	Chain of Custody Example
pH	8.6
Temp	21.1°C
DTW [86.1]	861
Depth must be within 20 feet of previous measurement.	
NOTE	Test successful

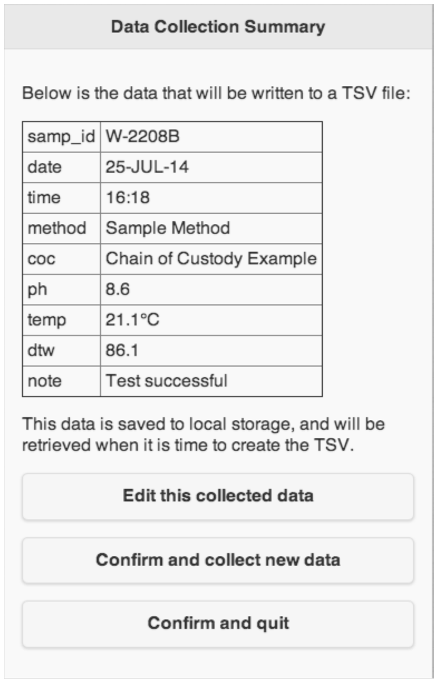
Finished collecting data

TEIMS Mobile App Builder

Fig 2. The data collection page in the built application.

The application has all of the input fields the user requested, which are generated using one React function call per input. Depending on the inputs, some of them may have extra functionality. Certain fields have autocompletion to make choosing from a list of hundreds or even thousands of items much simpler. Other fields, when selected, show well data that was most recently taken, giving the data collector a general context of what his or her data should be close to.

The application also features input validation. The rules are simple to edit so that data collectors themselves may specify the rules without in-depth programming knowledge. If a user does not enter a value into a required field, for example, the application displays a red error message underneath the corresponding field. At the bottom of the data collection screen, the user finds a "Finished collecting data" button. Upon clicking this, the validation plugin verifies that all inputs follow the specified rules. If any of the inputs do not adhere to the rules, the plugin will display an error message beneath each invalid input. If all inputs are satisfactory, Amplify will store all of the data that the user has entered so far, and the application will continue to the summary page.



Below is the data that will be written to a TSV file:

samp_id	W-2208B
date	25-JUL-14
time	16:18
method	Sample Method
coc	Chain of Custody Example
ph	8.6
temp	21.1°C
dtw	86.1
note	Test successful

This data is saved to local storage, and will be retrieved when it is time to create the TSV.

Fig 3. The summary page in the built application.

The idea of a summary page was suggested so that users can verify their work is 100% correct before they leave their well or other site of data collection. This is a precautionary step to catch any typographical errors. The user will have buttons for three options at this point. They can go back and edit their current information, in case they made a mistake. They can save their current data and reset the inputs, in order to collect data at the next location. Lastly, they may quit the application when they are finished collecting data.

One of the tools on ERD's applications server supports the uploading of tab-separated value (TSV) files to their environmental information database. When a TSV file is uploaded to this tool, the tool will automatically parse out the file's contents and prepare them for direct uploading to the database. The application built by the framework has a feature that allows the collected data stored on the device to be uploaded to the tool as a TSV file. This vastly simplifies the data collection process, as nothing needs to be printed and the user does not have to put in any effort to upload their collected data directly to the database.

IV. Evaluation

A formative evaluation will eventually be conducted, in which the framework will be released to a small group of data collectors. The data collectors will provide feedback on how the framework can be improved. After code adjustments have been made and extra features have been implemented, the formative evaluation will be reiterated to receive more feedback. This cycle will likely continue until the framework is at a high-enough standard of quality for data collectors to begin collecting data with the applications produced by it.

V. Future Work

Several functionalities are planned for future implementation. One of these is having the framework display a catalog of paper forms to the user, so that they may select the paper form they would use as a template for their application. This way, data collectors would not have to know every field of the form to accurately model it using the framework. Another feature is improving the code structure of the framework to be as modular as possible. The framework was designed to make each piece of code independent from the pieces around it, following good programming practice. However, there are some parts of the framework which were hard-coded due to time constraints. Lastly, a feature to be added is for a built application to be able to download fresh data directly from the database. Currently, the application builder will download the information and pass it through local storage to the built application. While this is a valid technique, if the user wants to use the application over the course of a few days, the database may be updated in the meantime, making the data they have on their device stale. If users could refresh their data right before they leave the office to collect data, the built application could be reused again and again over a considerable period of time.

Acknowledgments

I would like to thank Gary Laguna and the rest of the staff in the Environmental Restoration Department for providing great support throughout my internship. This was a phenomenal learning experience for me, and I am very grateful to have gotten the opportunity to work on this project in this department.

References

[1] Charland, Andre, and Brian LeRoux. "Mobile Application Development: Web vs. Native." *ACM Digital Library*. Communications of the ACM, May 2011. Web. 15 Aug. 2014.

[2] "Environmental Restoration Division." *Environmental Restoration Division*. Lawrence Livermore National Laboratory, 13 Apr. 2004. Web. 15 Aug. 2014.